A11102 970722

NBSIR 88-3784

NIST
PUBLICATIONS

MATERIAL HANDLING WORKSTATION IMPLEMENTATION

May 19, 1988

By:
Carl E. Wenger

QC
100
.U56
#88-3784
1988
c.2

U.S. DEPARTMENT OF COMMERCE National Bureau of Standards Gaithersburg, Maryland

NATIONAL INSTITUTE OF STANDARDS &
TECHNOLOGY
Research Information Center
Gaithersburg, MD 20899

NBSC
QC100
.U56
no.88-3784
1988
c.2

MATERIAL HANDLING WORKSTATION IMPLEMENTATION

Carl E. Wenger

May 19, 1988

This publication was prepared by United States Government employees as part of their official duties and is therefore a work of the U.S. Government and not subject to copyright.

Certain commercial software and hardware systems are identified in this paper to adequately describe the systems under development. Such identification does not imply recommendation or endorsement by the National Bureau of Standards, nor does it imply that the systems identified are the best available for the purpose.

Table of Contents

I. INTRODUCTION 1
1. PURPOSE OF THIS DOCUMENT 1
2. AUDIENCE 1

II. SYSTEM ARCHITECTURE 3
1. MATERIAL HANDLING FUNCTIONS 3
2. MATERIAL HANDLING EQUIPMENT 3
3. WORKSTATION CONTROLLER 5

APPENDIX A: AGV COMMAND AND STATUS MESSAGE FORMATS 13

APPENDIX B: DESCRIPTION OF TYPICAL COMMAND MESSAGE SETS 23

APPENDIX C: CHANGING AGV PROGRAM CODE OR DATA 26

APPENDIX D: PC COMPATIBLE AGV CONTROL SOFTWARE 32

APPENDIX E: ASRS COMMAND AND STATUS MESSAGE FORMATS 44

APPENDIX F: ROLLER TABLE COMMAND AND STATUS FORMATS 46

APPENDIX G: MATERIAL HANDLING SYSTEM WORK ELEMENTS 47

List of Figures

Figure 1. AMRF Shop Floor Layout

Figure 2. Material Handling Equipment

I. INTRODUCTION

1. PURPOSE OF THIS DOCUMENT

The purpose of this document is to provide a general description of the design and implementation of the AMRF Material Handling Workstation (MHWS). The MHWS equipment includes two Automatic Guided Vehicles (AGVs), an Automatic Storage and Retrieval Stystem (ASRS), and roller tables at other workstations. The material handling equipment with other AMRF equipment is shown in the AMRF shop floor layout, figure 1. The document should provide the reader with an understanding of concepts used to implement the MHWS.

2. AUDIENCE

The intended audience for this document is anyone who may be planning to build a material handling system for use in an automated facility.

3. CONTENTS FLOW

This document is one of a series of three MHWS documents. The two other documents in the series are "Material Handling Workstation, Recommended Technical Specifications for Procurement of Commercially Available Systems, NBSIR 88-3786" and "Material Handling Workstation Operator Manual, NBSIR 88-3785".

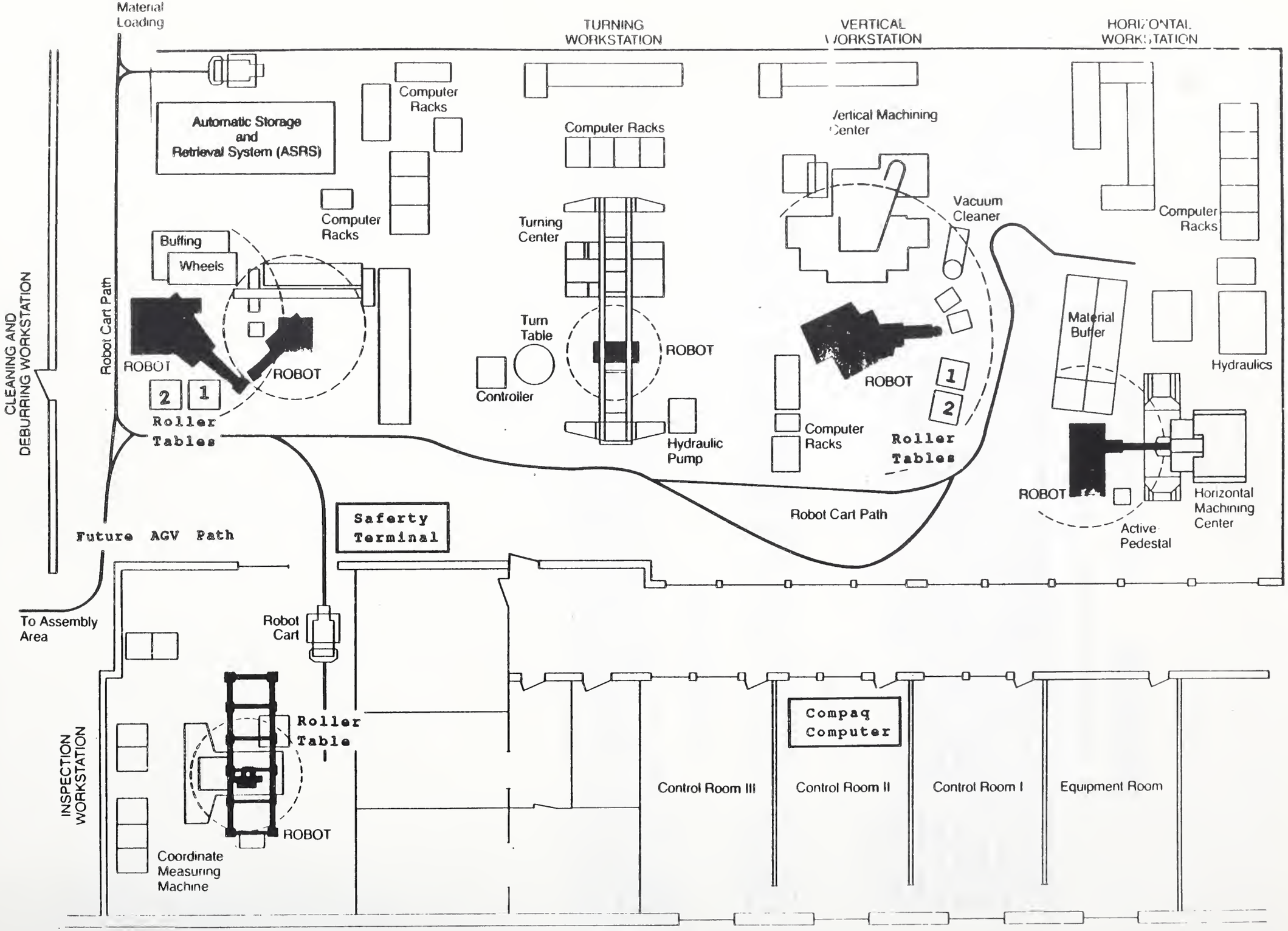


Figure 1. AMRF Floor Layout

II. SYSTEM ARCHITECTURE

1. MATERIAL HANDLING FUNCTIONS

The AMRF Material Handling Workstation is composed of (1) two automatic guided vehicles (AGVs) that transport trays of parts and tooling between the workstations, (2) an automatic storage and retrieval system (ASRS), and (3) roller tables at other workstations onto which trays are transferred. The MHWS controller manages these subordinate systems by executing the operations specified in process plans. The controller also provides interfaces that are required to integrate material handling with other AMRF systems. The configuration of the material handling equipment is shown in figure 2.

2. EQUIPMENT OF THE MATERIAL HANDLING WORKSTATION

2.1 Control Computer

The material handling control computer is a COMPAQ PORTABLE 286. Figure 1 shows how the control computer is connected with the material handling equipment. It executes a control program written in the 'C' programming language which controls the activities of the material handling equipment. It communicates with all the material handling equipment through hard wire RS232 connections sending commands to and receiving status from this equipment. It receives commands from the Cell control computer and reports status to the Cell.

2.2 Automatic Guided Vehicles

The material handling equipment includes two Automatic Guided Vehicles (AGVs) which are used to transfer parts trays between the Automatic Storage and Retrieval System (ASRS) and the work stations. Each AGV contains an 8085 control computer which communicates with the MHWS control computer via inductive wire buried in the shop floor. A second wire in the floor provides a guidance signal for the AGV to follow.

The AGVs are controlled by the material handling control computer. Seven byte command messages are transmitted to the AGVs and seven byte status messages are received. The formats of the AGV command and status messages are detailed in Appendix A.

The AGV command messages are combined into message sets of 1 or more messages which are transmitted to the AGV in sequence. The received command messages are stored in a queue in the AGV computer until the last message of a message set is received, and then all received messages are executed. After execution of all

Equipment of the AMRF Material Handling Workstation

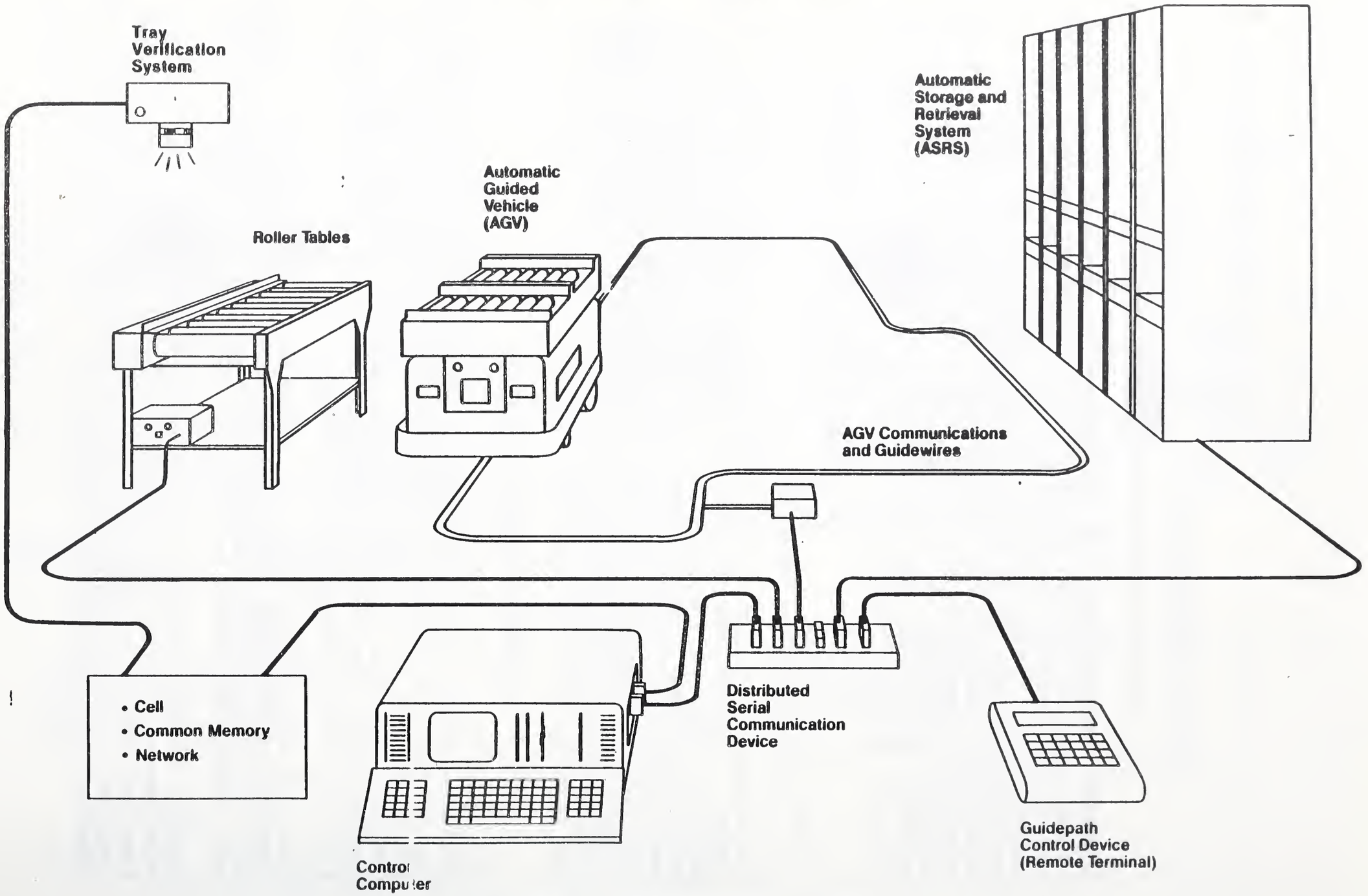


Figure 2. Material Handling Equipment

received messages is complete, the AGV stops and awaits further instruction. A message set usually completes one AGV function such as traveling from one tray transfer point to another, or transfering a tray to or from the AGV. Typical message set functions are detailed in Appendix B.

Both of the AGV systems were originally identical. They contained an on board AGV control system which included an 8085 central processor. The control program is stored in eight programmable read only memory (PROM) devices. A maximum of 32K of memory is used for the control program. The control program is implemented in the Forth programming language. Any modifications to the control program must be made by programming new PROM chips. A typical procedure to program new chips is given in Appendix C.

At the present time, the AGV control system hardware and software of one of the two AGVs are being upgraded. The new hardware will be controlled by an 80286 CPU and is IBM PC compatible. Off-the-shelf analog-to-digital and digital-to-analog boards are being used. The control software is being replaced and the new software is implemented in the Turbo "C" programming language. Because of the PC compatibility, most of the control software development may be done on standard office PC computers. Preliminary documentation on the new AGV software is given in Appendix D.

2.3 Storage and Retrieval System

The Automatic Storage and Retrieval System (ASRS) is a KARDEX 8000 Industriever Storage Retrieval System. The system has six identical modules with each module storing 33 parts-trays on shelves in the module. Each module has a conveyor for transferring trays between the module and an AGV. The system has random access to the trays in a specific module in that a tray may be moved between any shelf in the module and the conveyor attached to that module. To access all trays in the system would require an AGV stop at each module.

The MHWS control computer communicates with the ASRS through one RS232 hard wire connection sending commands to the ASRS and receiving status upon request. Each command contains a module id byte, and a command byte. Commands which transfer trays between the conveyor and a shelf also contain the shelf number. The command and status message formats are given in Appendix E.

2.4 Roller Tables

Each workstation (except the Horizontal Workstation) served by the material handling system has 1 or 2 roller tables which are used to transfer trays between an AGV and the workstation. The

tables are manufactured by the Litton Corp. and the table controllers were designed and constructed in-house at NBS. The controllers accept 1-byte commands to control the electric motors which rotate the rollers to move trays on and off the tables. Each table controller controls 1 or 2 tables and is connected to the MHWS control computer through a hardwire RS-232 connection. Upon request, the controller will return a 6-byte status message. The command and status message formats for the table controllers are given in appendix F.

3. WORKSTATION CONTROLLER

3.1 General Description

The MHWS controller is a software system developed at NBS which executes material handling commands received from either the Cell controller or a MHWS operator. The MHWS controller (MHWSC) operates in two primary modes, stand-alone or remote. While operating in the remote mode, commands are received from the Cell, and while in the stand-alone mode, commands are received from the MHS operator. In either mode, the commands are decomposed into low level commands to be executed by the equipment. The MHSC is executed on a Compaq computer which has a hard wire serial link to each equipment controller.

The MHSC executes the Cell work element DELIVER_TRAY by executing material handling process plans containing equipment level work elements GOTO, GOTHRU, GET_TRAY, PUT_TRAY, and TRANSFER_TRAY. Work elements GOTO and GOTHRU are decomposed into AGV command messages to send the AGVs to the tray transfer points. Work element GET_TRAY gets a tray from an ASRS shelf and transfers it to the conveyor for loading onto an AGV, while PUT_TRAY puts a tray off-loaded from an AGV back onto an ASRS shelf. Work element TRANSFER_TRAY transfers trays between an AGV and either the ASRS or a roller table. Detailed descriptions of the material handling work element formats are given in Appendix G.

All of the MHWS work elements listed above can also be initiated by an operator of the material handling control computer. In addition to the DELIVER_TRAY work elements, several additional work elements can be initiated by the MHWS operator. The CHARGE_BATTERY work element extends the charge probe, charges the battery for a specified period of time, and then retracts the probe after the specified time has elasped. The RETRACT_PROBE work element allows the charge probe to be retracted immediately whether or not the specified charge cycle has been completed.

The software modules of the MHWS controller include: a system supervisor which sequences the activation of major subsystems, a

communications manager which handles message traffic, a data manager which handles the translation of data to/from internal formats from/to external formats, a material handling manager that sequences equipment level instructions to the material handling equipment, and a screen manager which handles the user interface.

3.2. System Supervisor

This module is responsible for initializing the MHWS system, coordinating orderly system shutdowns, and activating each of the following major system modules during normal operations:

- Communications Manager
- Data Manager
- Manufacturing Manager
- Material Handling Manager
- Screen Manager

All of the above modules are described in greater detail in the paragraphs that follow. The System Supervisor activates each module at least once per control cycle. Running on a COMPAQ PORTABLE 286, the system executes several control cycles per second.

During each control cycle, the System Supervisor invokes a subroutine which advances the system clock and updates counters and timers. Next, it invokes the Communications Manager in RECEIVE mode, to give it the opportunity to handle incoming communications traffic. Then, the Data Manager is activated in INPUT mode to translate any newly received mailgrams into internal data formats.

The Material Handling Manager is activated to check for new orders from the Cell and to monitor status of the MHWS equipment. When equipment status so indicates, new MHWS process plan work elements are decomposed into equipment instructions and transmitted to the equipment.

After the Material Handling Manager completes its processing, the System Supervisor activates the Data Manager in OUTPUT mode. The Data Manager checks internal data structures for changes and generates mailgrams in prescribed external formats, as required. Finally, the Supervisor invokes the Communications Manager in SEND mode. The Communications Manager checks to see if it is time to transmit messages, then looks for new outgoing mailgrams. It sends them across a serial link to a Sun computer which provides common memory mailboxes which are accessed by the AMRF network. Between each entry to a major module described above, the Screen

Manager is activated. The Screen Manager determines if any data needs to be updated on the monitor, and refreshes it. The System Supervisor loops and continues to repeat the entire sequence until it is shutdown.

The System Supervisor is able to repeat this entire process quickly because events that require any significant processing occur only at sparse instants over time. Furthermore, all major data elements within the system have tags, called "data sequence numbers" or DSNs, associated with them. The DSN is an array of counters. One counter is allocated to the data element itself. Another counter is allocated for each major module identified above. Whenever the contents of a major data element are changed, the self counter is incremented by the system that changed the data. During the control cycle, each module that has a reason to look at the data element, compares its own module counter to the self counter. If they differ, it processes the data and sets its own counter to match the data self counter. If they are the same, the system has already processed the data and it may continue with its next activity. This scheme minimizes unnecessary processing of data by ensuring that each piece of information is only handled by each subsystem once.

3.3. Communications Manager

The Communications Manager coordinates the transmission and receipt of all message traffic for the MHWSC. It is implemented as a finite state machine. When the module is activated by the System Supervisor, it cycles through the state machine performing communications functions until an EXIT state is reached. Upon each entry to the Communications Manager, it first determines whether or not it is time to send or receive messages. Message transmission is currently set to occur on 7.5 second intervals for performance reasons. Because of the 9600 baud serial link to the Sun computer and the relative infrequency of traffic at the MHWSC level in the AMRF hierarchy, this interval seems to be satisfactory. When a direct network link is established on the Cell, traffic will probably be processed in each control cycle.

If it is time to transmit messages, one of the following actions is taken. If the Communications Manager is activated in SEND mode, it checks the counter, i.e. DSN, on each mailbox to determine whether or not it contains new outgoing mail. Each mailbox has a pointer to the chain of ready message blocks that make up the mailgram. Each message block has a 4 byte binary header and between 1 and 240 bytes of text. The header includes a checksum, a block length, a mailbox number, and the number of the particular block within the mailgram. Pointers to all READY message blocks are entered into a READY blocks table. Through a

handshake protocol, the Communications Manager places the state machine-based communications server on the Sun microcomputer in RECEIVE mode. Message blocks are subsequently transmitted and acknowledged. The servers at both ends of the serial link automatically handle some error recovery and retransmission of garbled messages. From the mailboxes internal to the server on the Sun, the messages are transferred to the Sun common memory areas by subroutine calls. Once mailgrams reach this memory area they are accessible to the AMRF network and all other systems within the AMRF that are connected to the network.

If the Communications Manager is in RECEIVE mode, it uses the same protocol to place the server running on the Sun in SEND mode. The other communications server follows a procedure similar to that outlined above to transfer new mailgrams that it has obtained from common memory on the Sun. As each message is received on the PC, the Communications Manager obtains a data block from a free list, copies the incoming bytes into the block, reads the header, performs checksum calculations, chains error-free blocks into the specified mailbox, and updates the appropriate data sequence numbers. If necessary, it will request retransmission of garbled blocks.

It is possible that network services will be integrated with common memory on the PC some time within the next year. When this changeover occurs, only the operation of the Communications Manager module will be affected. The communications function is transparent to all other modules within the MHWS control system.

3.4. Data Manager

The principal functions of this module are the translation internal data to and from external message formats, and the management of updates of internal and external data areas. Currently, the Data Manager handles the translation of process plans, commands, and status information. Work is currently underway to support a wide variety of data base transactions and reports.

When the Data Manager is activated in INPUT mode, it checks the DSN on each mailbox to see if a new mailgram has arrived. If new mail is found, it checks the type code associated with the mailbox, and activates the parser that handles that type of message traffic. The parser reads the message, performs required translations, and enters the information into the appropriate internal data areas. DSNs on the mailboxes and internal data structures are updated as required. When all mailboxes have been handled, the Data Manager returns execution back to the System Supervisor.

When the Data Manager is activated in OUTPUT mode, it checks the DSN on each internal data element that it monitors to see if a new mailgram must be generated. If new outgoing command or status data is found, the appropriate message generation software is activated. The message generation software determines which mailbox is affected, returns any message blocks which are chained to that mailbox to the free list, obtains new blocks as required from the freelist for storing the new message, enters appropriate header information into each message block as it is used, chains the message blocks of the new mailgram into the mailbox, and updates the appropriate DSNs on the mailbox and the internal data structure.

The Data Manager provides a layer of transparency between the Communications Manager and the Manufacturing Manager. External message formats can be modified without affecting the functioning of the Manufacturing Manager and internal manufacturing data can be restructured without affecting the Communications Manager.

3.5. Transition Manager

The Transition Manager module has not been implemented in the current version of the Material Handling Controller. When this module is implemented, it will manage the transition states during startup and shutdown of the MHWS.

3.6. Material Handling Manager

This subsystem processes material handling orders to transfer trays between the storage and retrieval system and the work-stations. Orders may be entered into the system by three different methods, (1) command from the Cell, (2) manual entry by MHWS operator, or (3) automatic entry from an orders file. The communications manager must be activated to receive orders from the Cell. Each order for tray delivery specifies a process plan id and a tray id. Orders from the Cell may be received only if the communications manager is activated and communications with the Cell is established. The MHWS operator may enter an order at any time using the order entry screen. Up to 100 orders may be read in from the orders file and all orders in the file may be cycled up to 100 times. For all order entry methods, the orders are processed in sequence as received unless the workstation involved is not ready to receive a tray. In that case, the order is temporarily placed on hold for later processing.

At the beginning of each control cycle, the system checks to see if a new material handling order has been received. If so, the order is placed in a queue and processing continues on the

current order. When the current order is completed, the entire queue is scanned and each order status is checked. If an order status of not DONE is found, the status of the specified workstation is checked to determine if it is ready to receive a tray. If the workstation is ready, the order is analyzed and the specified process plan is loaded into memory from a local disk file. Each order specifies a delivery tray id as well as the process plan id. The current locations of the tray to be delivered and the tray to be returned are retrieved from the local data base, and the process plan is edited so that the plan contains actual storage and delivery locations and tray id's. Each work element in the plan is then executed in sequence.

After processing of a work element is initiated, the next work element of the plan is analyzed to determine which equipment is needed to complete the task specified in the work element. If the status of any necessary equipment is BUSY, that equipment status is again polled. After the status of all necessary equipment is found to be READY, the "next work element" becomes the current work element and commands are sent to the equipment to complete the task.

If the current work element is a GOTO or GOTHRU, an AGV is the only equipment required. These work elements specify that the AGV travel over a path segment between two adjacent transfer points. The beginning and ending transfer points (TO and FROM locations) are decoded from the plan and are matched with table entries from data file "msgset.dat". When a record from the file is found which matches the TO and FROM locations, the data set number is extracted from the record. The individual AGV commands in that data set are then transmitted to the AGV.

The MHWS controller processes the GOTHRU work elements somewhat differently from GOTO elements. A series of GOTHRU's may be processed to send an AGV from any transfer point on the AGV path to any other point. A series of one or more GOTHRU's in a process plan must always be terminated by a GOTO work element. A GOTHRU message set is actually a modified GOTO message set. The message numbers in the GOTO message set commands are modified so that the command messages are presented to the AGV in numerical sequence. Also, the last message in each message set is modified to specify that more messages follow.

The GET_TRAY and PUT_TRAY work elements always require only the ASRS as an equipment resource. GET_TRAY gets a tray from the specified shelf and places it on the stub conveyor for later transfer to an AGV. PUT_TRAY returns a tray from the conveyor to the specifed shelf. Since GET_TRAY and PUT_TRAY only involve the

ASRS and because GOTO involves only an AGV, these work elements may be processed simultaneously.

The CHARGE_BATTERY work element is processed to charge the AGV batteries when the AGV is at the charging station. Usually, it is included in a plan to go to the charging station from the ASRS. Currently, this work element is initiated by the MHWS operator with the MHWS controller operating in its stand-alone mode. When the new PC compatible hardware is in operation, the AGV status data will include the battery voltage so that battery charging can be done automatically when necessary. This work element has a charge time argument. When the batteries have been charged for the specified time, the remaining portion of the plan is processed. Usually, this part of the plan would include work elements to retract the charge probe followed by GOTHRU's and a GOTO to return to the ASRS. The RETRACT_PROBE may be executed at any time whether or not the specified charging cycle has been completed.

3.7. Screen Manager

The Screen Manager's primary responsibility is handling the user interface. The display and menu interface for the MHWS control system uses techniques found in many PC-based applications, such as spreadsheets and database systems. The display screen is divided into three areas: 1) the menu, 2) the data page, and 3) the status bar. These areas are described below.

Most user interaction with the system is directed through the menu area. The menu area occupies the topmost three lines on the monitor screen. The first line displays the name of the menu that the user has currently selected. The second line provides selection operations or data that may be chosen by the user. Left and right cursor control keys move the highlighting bar over the possible selections. Depressing the ENTER key, makes a selection which may cause an associated menu function to be called and/or another level of descent in the menu structure to occur. Depressing the ESCAPE key, causes the user to back up a level in the menu structure. Other keys provide the user with direct access to many functions and screens, too numerous to describe here. The third line displays the next level of options with the selection that is currently highlighted on line 2.

Most information about the MHWS is presented in the second area on the screen, the data page. The data page covers from line 4 to line 24 on the screen. Many different page formats are defined which provide control and communications information to the user in real-time.

The last area on the screen, the status bar, provides summary or diagnostic information. The highlighted status bar displays the current clock cycle, the current screen identifier, communications status, and the current time. It provides constant feedback to the user that the MHWS is operating correctly.

Excluding the initialization and shutdown phases, the Screen Manager is always activated in UPDATE mode. In UPDATE mode, it first determines if a new screen has been selected by the user. If a new screen is selected, the data area is cleared, and the new background/foreground fields are displayed. The Screen Manager next determines if the new screen makes use of cursor control and other keys to change display data. If these keys are to be activated, the menu area is cleared, i.e. turned off. Next, the keyboard input buffer is checked, and if keystrokes are present, they are processed.

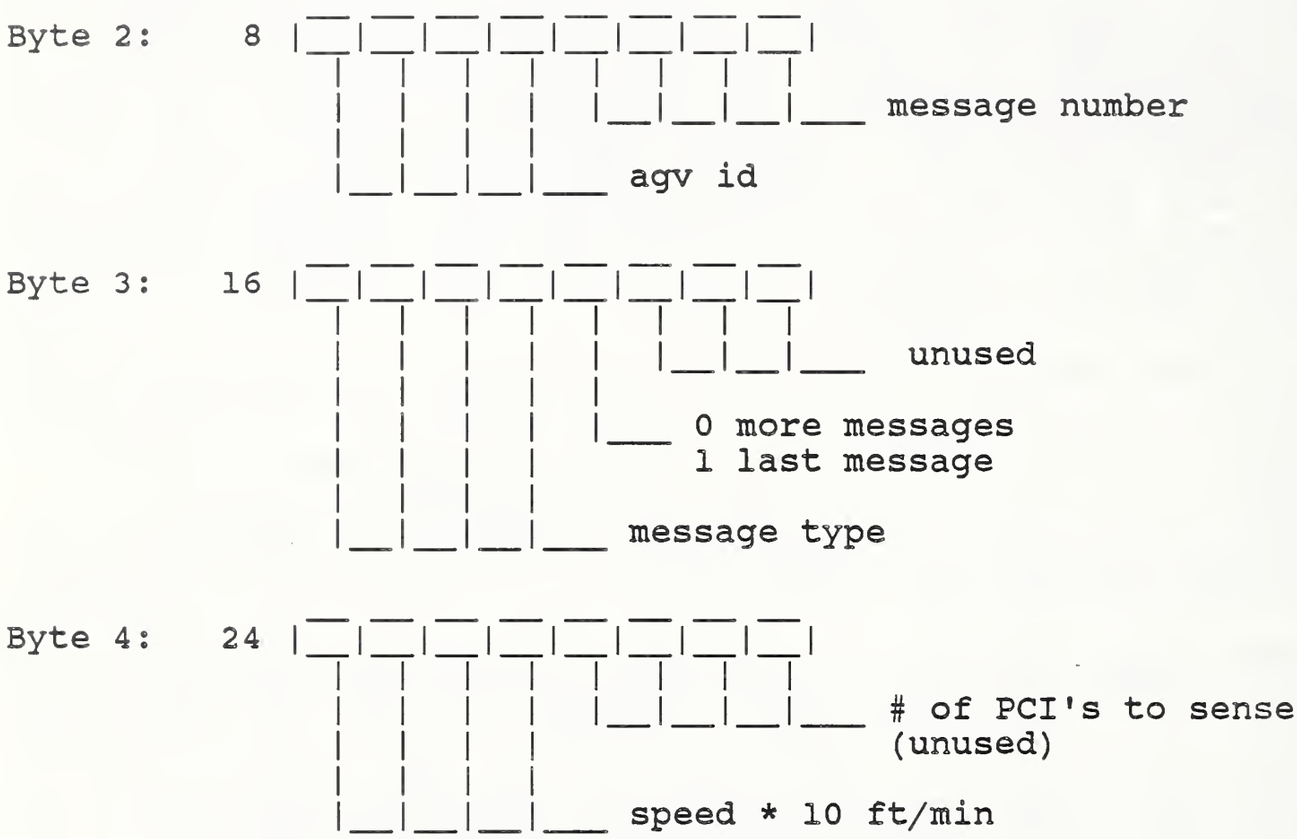
If the Screen Manager finds that a new screen has not been selected, it performs a series of tests and checks on DSNs to determine if the foreground data fields on the screen need to be refreshed, and takes appropriate action. Depending on the functions which have been defined for a particular screen, keystrokes may affect the data that is displayed or may cause a return to the menu mode of operation.

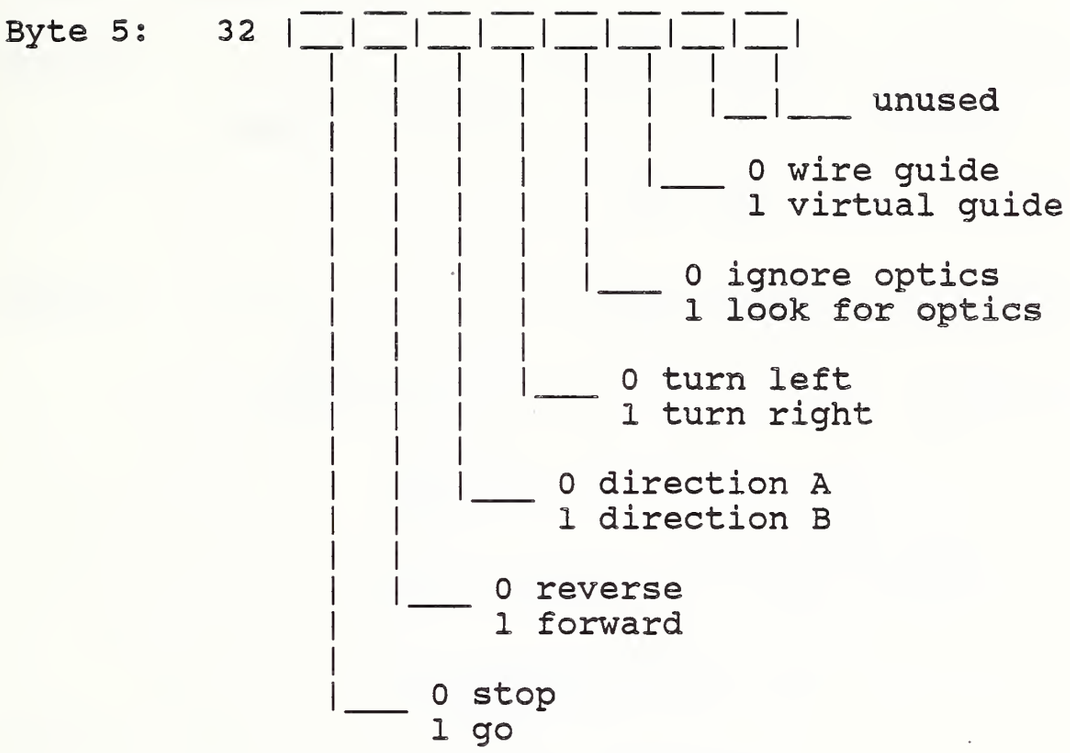
The user interface system coordinated by the Screen Manager, facilitates the development of new screens or the modification of existing ones. More than a dozen different screens are currently available in the system. Each screen in turn may have several different data pages associated with it. Because of the low overhead associated with checking DSNs, system programmers may invoke the Screen Manager at anytime to have the screen refreshed, with virtually no effect on system performance. The current implementation of the MHWS control system activates the Screen Manager many times during each clock cycle.

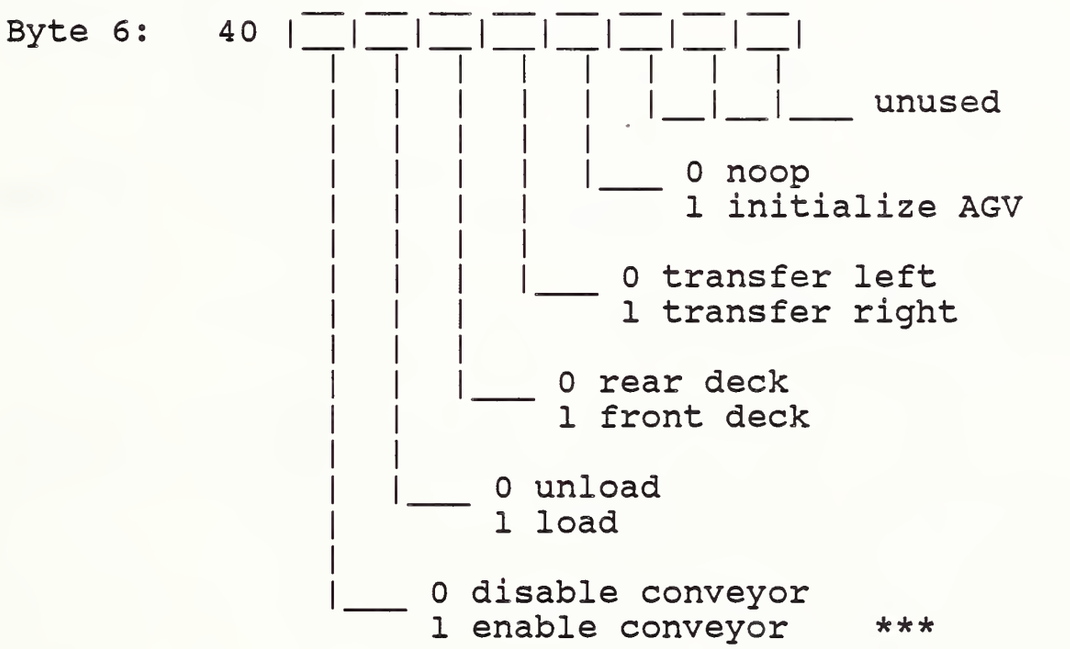
Appendix A. AGV Command and Status Message Formats

Note: For all messages, Byte 1 contains the value EE hex, and byte 7 is the checksum of bytes 2 through 6.

Type 0 Command Format

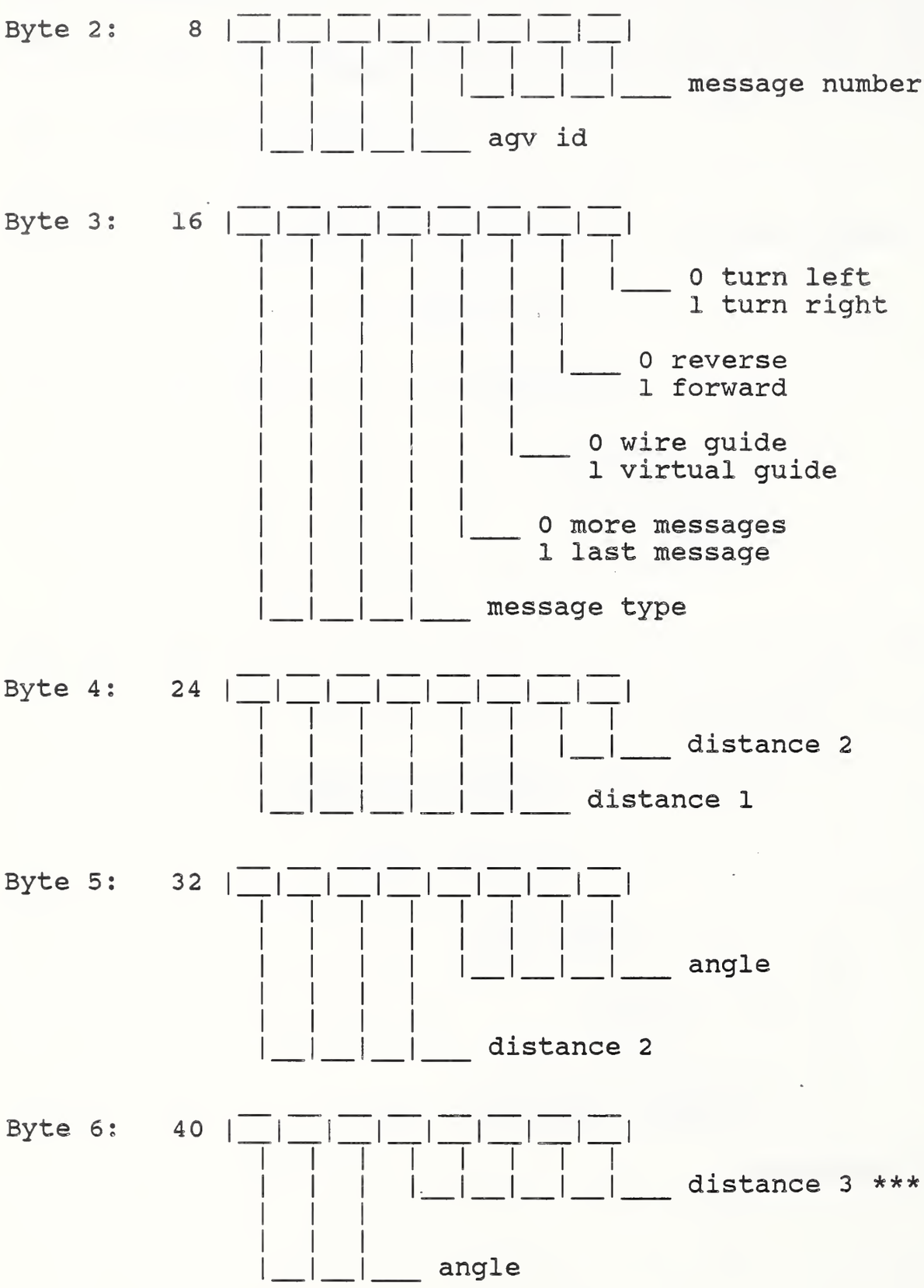




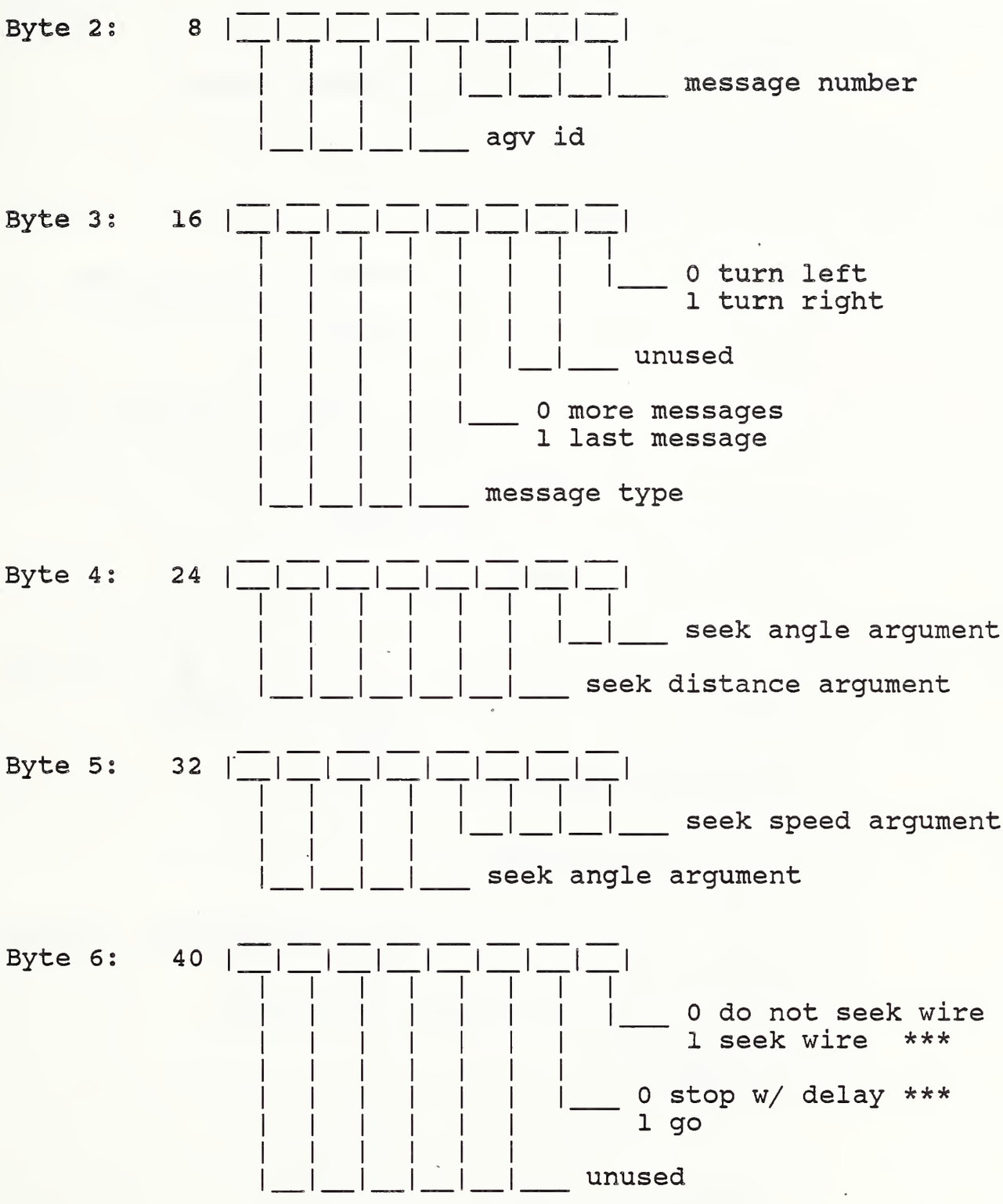


*** Command terminator

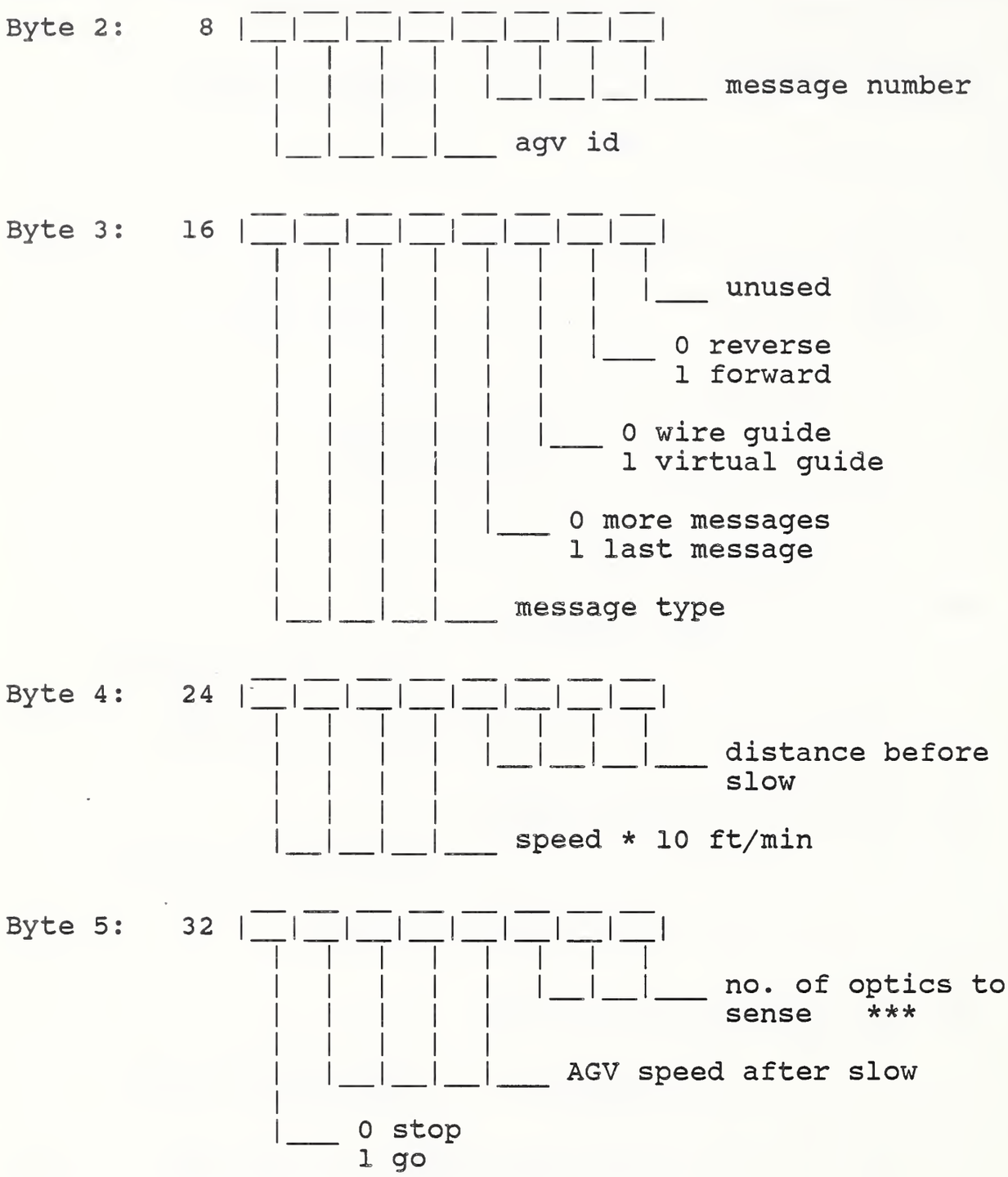
Type 1 Command Format

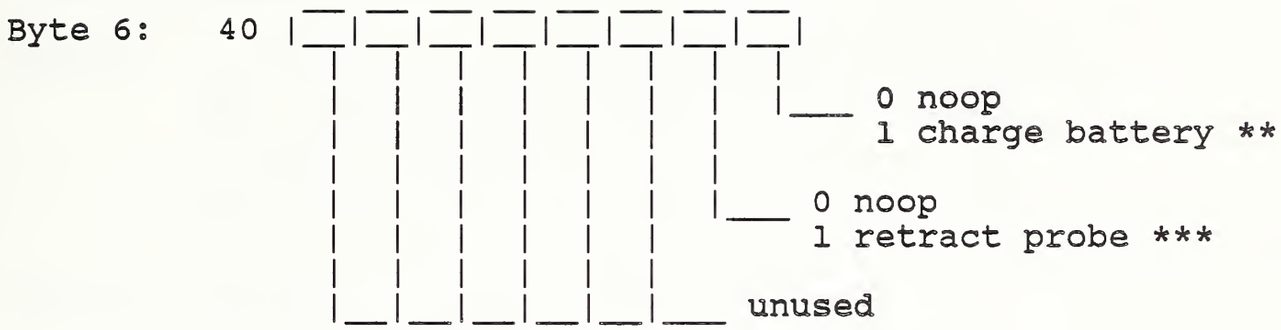


Type 2 Command Format



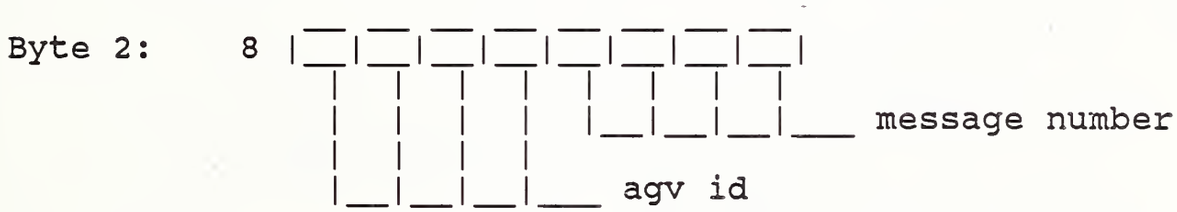
Type 3 Command Format

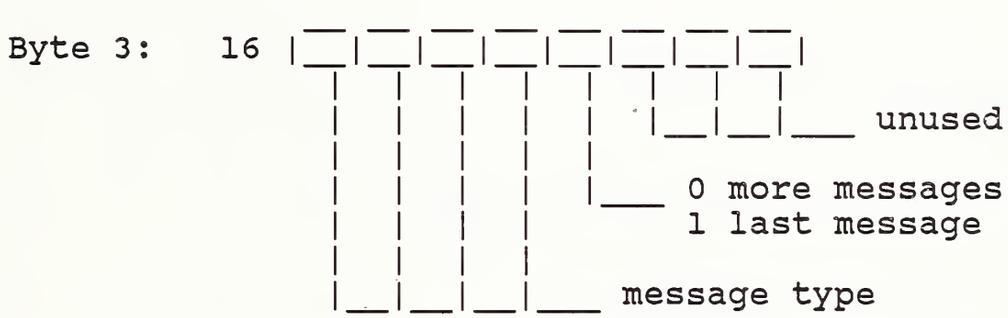


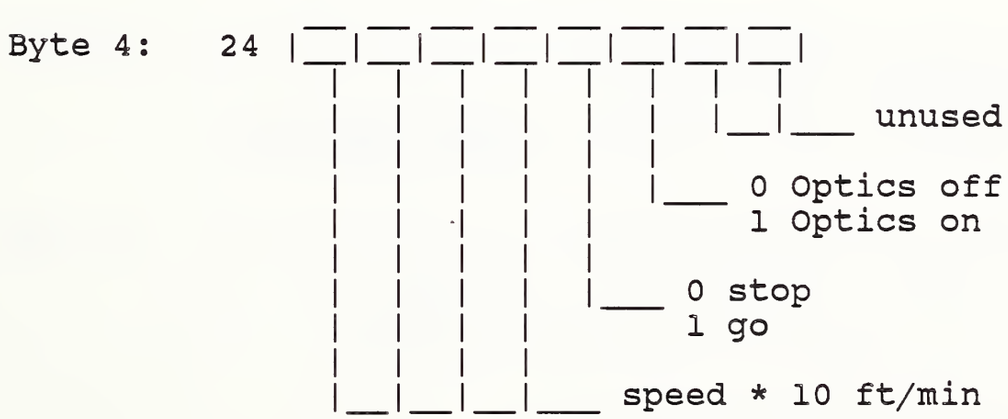


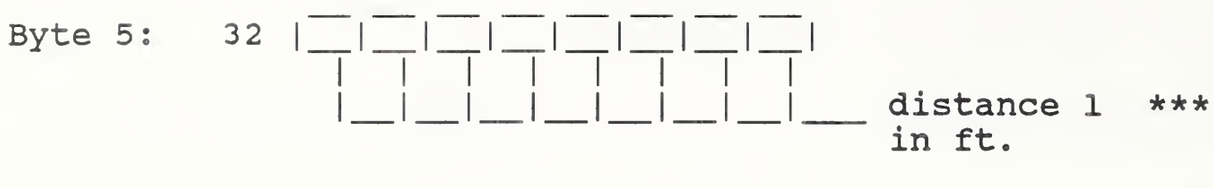
** End command sequence
*** Command terminator

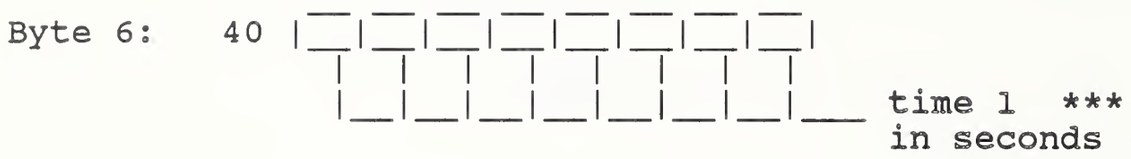
Type 4 Command Format





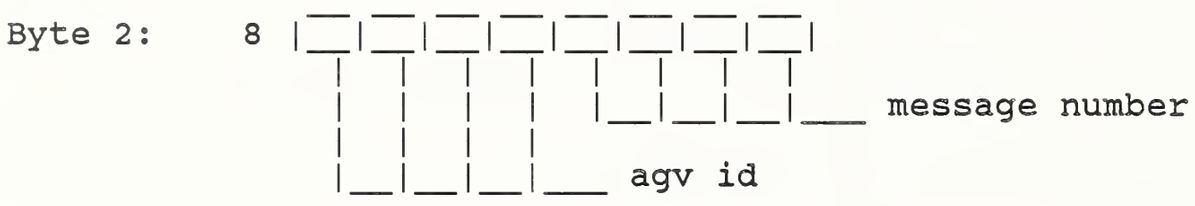


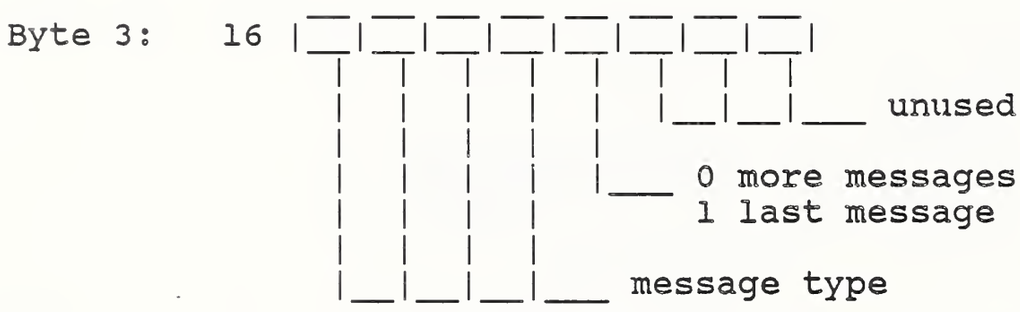


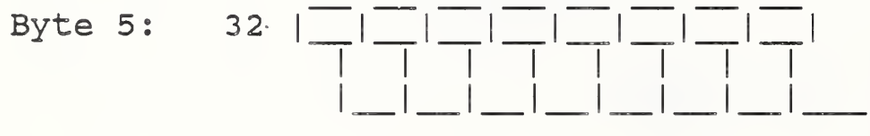


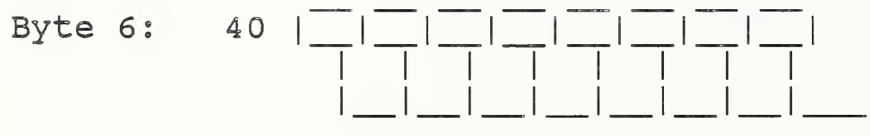
*** Command terminator

Type 5 Command Format

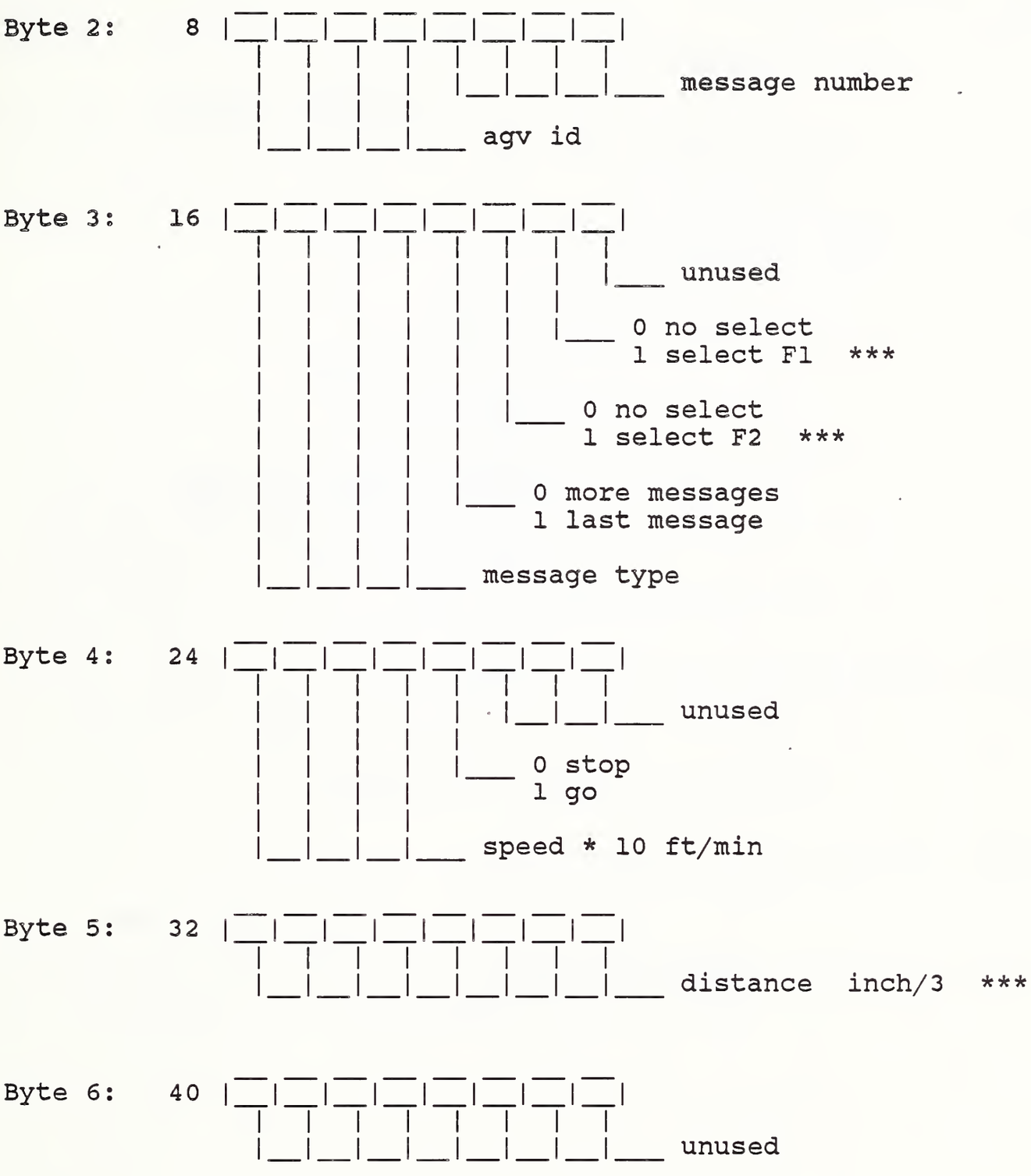




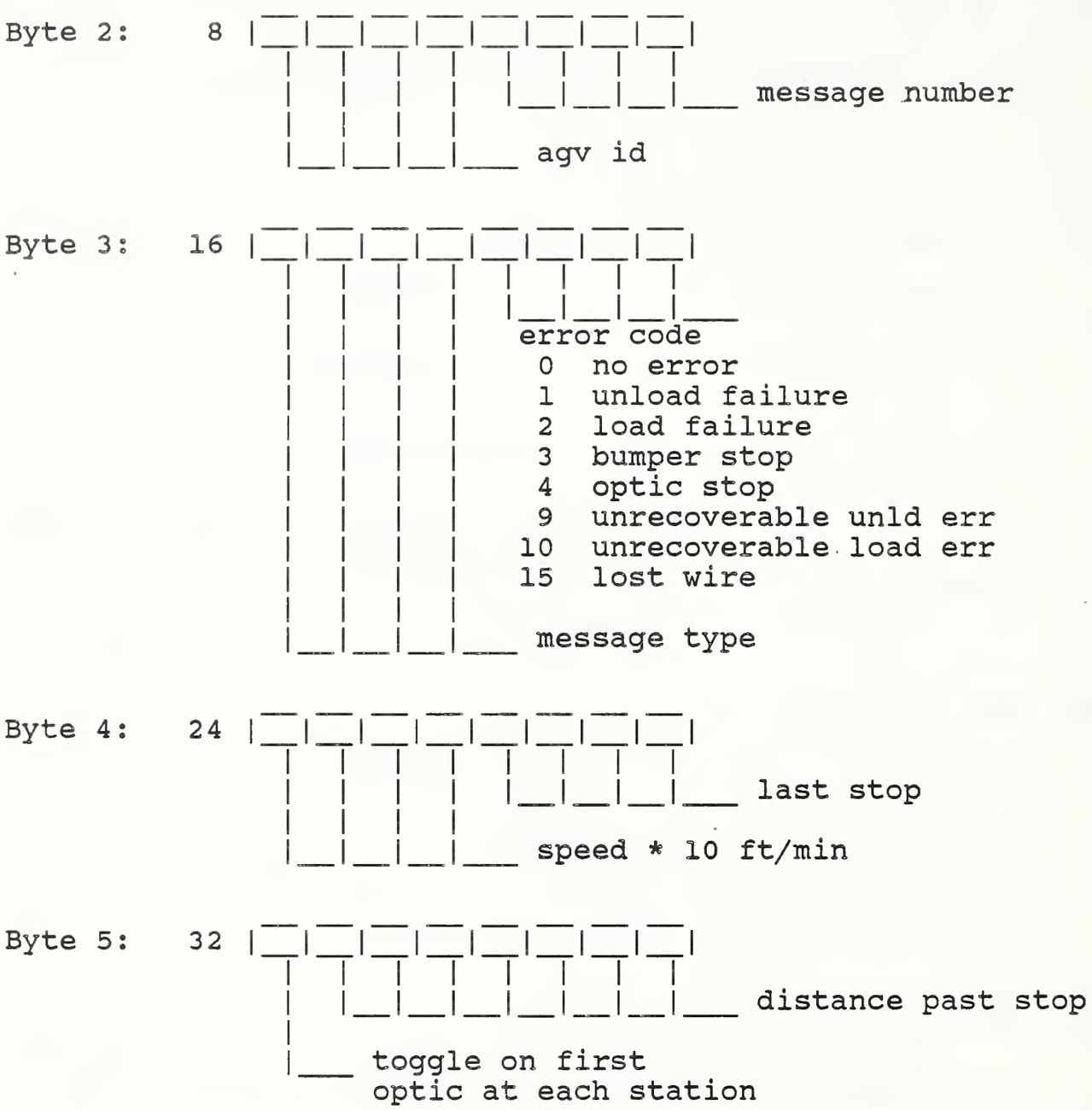




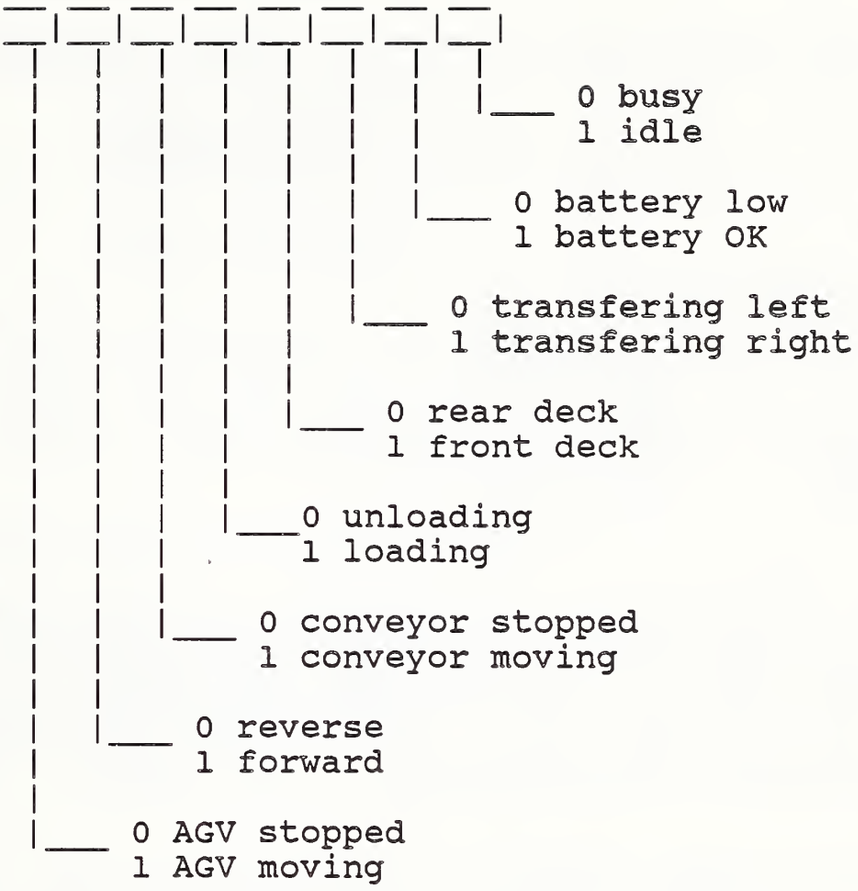
Type 6 Command Format



Type 14 Status Format



Byte 6: 40


0 busy
1 idle
0 battery low
1 battery OK
0 transfering left
1 transfering right
0 rear deck
1 front deck
0 unloading
1 loading
0 conveyor stopped
1 conveyor moving
0 reverse
1 forward
0 AGV stopped
1 AGV moving

Appendix B. Description of Typical Command Message Sets

Following is a description of an AGV command message set used to implement either the GOTO or GOTHRU work element.

Message set 0: Go to CDWS table 1 from ASRS unit 1.

Command Message 0: EE 00 14 35 25 05 73

Message Parameter Values

AGV ID: 0	Turn Left
Message No: 0	Distance 1: 3.25
Message Type: 1	Distance 2: 4.50
More Messages	Turn Angle: 40
Virtual Guide	Distance 3: 1.25
Reverse	

Pattern turn command with more messages to follow. Move without guidance 3.25 feet straight backwards. Move back 4.50 feet with steering angle set to 40 degrees left. Set steering to 0 degrees and move back 1.25 feet. Pass control to the next command in the queue when the pattern turn is completed. A default speed of 40 fpm is used for all reverse moves.

Command Message 1: EE 01 24 00 A3 04 CC

Message Parameter Values

AGV ID: 0	Seek Distance: 0
Message No: 1	Seek Angle: 10
Message Type: 2	Seek Speed: 30
More Messages	Stop with Delay
Turn Left	Don't Seek Wire

Type 2 command with more messages to follow. Stop AGV immediately after execution is started and wait 2 seconds before passing control to the next command in the Queue. When the "stop with delay" action is specified, the turn direction, distance, angle, and speed parameters have no effect on the AGV action.

Command Message 2: EE 02 25 04 53 07 85

Message Parameter Values

AGV ID: 0	Seek Distance: 1
Message No: 2	Seek Angle: 5
Message Type: 2	Seek Speed: 30
More Messages	Start AGV (Go)
Turn Right	Seek the Wire

Type 2 command with more messages to follow. Set steering angle to 5 degrees left and speed to 30 feet per minute. Move 1 foot (the seek distance) and then start seeking the wire. If the wire is not found within a default distance of 3 feet, the AGV stops.

Command Message 3: EE 03 45 CC 0A 00 1E

Message Parameter Values

AGV ID: 0	Start AGV (Go)
Message No: 3	Stop Optics On
Message Type: 4	Distance 1: 10
More Messages	Time: 0
Speed: 120	

Type 4 command with more messages to follow. Travel 10 feet at 120 feet per minute and then pass control to the next command in the queue.

Command Message 4: EE 04 45 CC 0A 00 1F

Message Parameter Values

AGV ID: 0	Start AGV (Go)
Message No: 4	Stop Optics On
Message Type: 4	Distance 1: 10
More Messages	Time: 0
Speed: 120	

Type 4 command with more messages to follow. Travel 10 feet at 120 feet per minute and then pass control to the next command in the queue.

Command Message 5: EE 05 3B AE 63 00 51

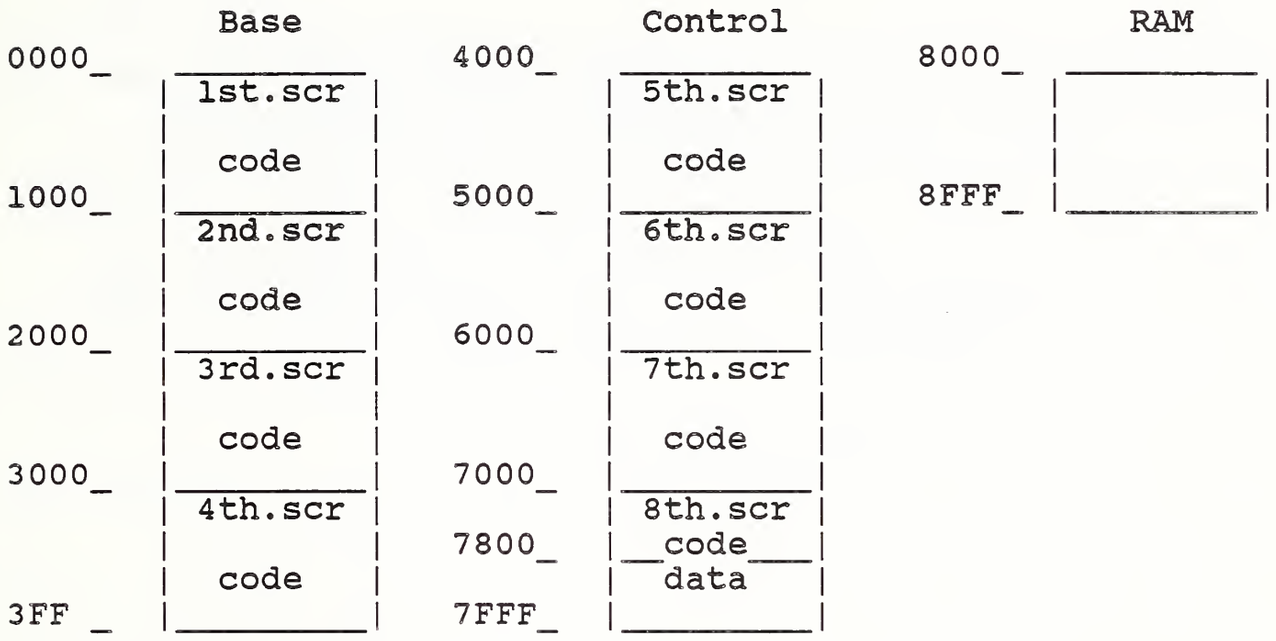
Message Parameter Values

AGV ID: 0	Ft. Before Slow: 14
Message No: 5	Stop AGV
Message Type: 3	Speed 2: 120
Last Message	Optics to Sense: 3
Follow Wire	Don't Retract Prb.
Forward	Don't Charge Batt.
Speed 1: 100	

Stop at optic command, the last message in the set. Follow the wire for 14 feet at 100 fpm then change speed to 120 fpm if going through or the default speed of 12 fpm if stopping. Control is passed or the AGV stops at the 3rd optic. When a message set is used for the GOTHRU work element, the "Stop AGV" bit is changed in the last message of the message set.

Appendix C. Changing AGV Program Code or Data

AGV ROM Memory Map



Changing data in ROM

Most of the command data sets used to implement the AGV work elements are transmitted to the AGV from the MHWS controller at the time of execution. However, some of the longer message sets are stored in the read only memory (ROM) devices to save time in transmitting commands to the AGV. These special message sets are stored in the 8th ROM device which occasionally must be reprogrammed when shop floor conditions change. The message set data is stored beginning at memory location 7800 hex. The process for modifying this data is as follows:

1. Enter the MHWS controller in mode 1
2. Find the work element and make the change.
3. Scroll through the elements you want in ROM.
4. Exit from the controller.
5. Copy CMD.DAT into the FORTH directory.
6. Using debug or Forth, copy CMD.DAT into file 8th.scr starting at the second 2K boundary.

7. Burn a PROM from 8th.scr as described below.

Description of Process to Change Forth Code or Data

This description is intended to provide the information neccessary to generate ROM based object code for the NBS AGV. It is assumed that the user has sufficient knowledge of Forth, process control, and guided vehicles to modify the source listing. Screens are presented as they should appear to the user.

<CR> Carriage Return -> User Input Required
<F1> F1 Function key
C> System Prompt

Sample Session

I. Compile

A. Base:

```
->      C> mc

        LMI Metacompiler 2.0
        Host: IBM PC  Target: 8080  Created 02/20/85
        Copyright (c) 1985 Laboratory Microsystems Inc.

->      Source file name?          [ ]          NBSBASE
->      Intermediate files?        [MC]         <CR>
->      Defining word prepass?     (Y/N)  [N]   <CR>
->      Printer error audit?       (Y/N)  [N]   <CR>
->      Print symbol table?        (Y/N)  [N]   <CR>
->      Pause on errors?           (Y/N)  [Y]   <CR>
```

B. Control:

```
->      C> mc

        LMI Metacompiler 2.0
        Host: IBM PC  Target: 8080  Created 02/20/85
        Copyright (c) 1985 Laboratory Microsystems Inc.

->      Source file name?          [ ]          NBS
->      Intermediate files?        [MC]         NBSBASE
->      Defining word prepass?     (Y/N)  [N]   <CR>
->      Printer error audit?       (Y/N)  [N]   <CR>
->      Print symbol table?        (Y/N)  [N]   <CR>
->      Pause on errors?           (Y/N)  [Y]   <CR>
```

II. Locate

```
->      C> loc nbs
```

III. Burn Proms (using Unipak eprom programmer)

```
->      C> pl

                PROMlink
    Programmer/Computer Communications Link
                Version 2.01

       Copyright Data I/O Corp. 1984, 1985


Contact with programmer established.

->      <F1>
```

```
Title: User name and Job number
Programmer: Model 29 with UniPak                Programmer size: 16K
Device type: AMD 2732                           I/O Format: Binary
Port: COM1/9600 baud                            User Function File: *
Data File: *                                    Directory: c:\FORTH

          (1) Program device                    (6) Error logging
          (2) Verify device                     (7) File functions
          (3) Load RAM from device              (8) Edit functions
          (4) Load RAM from file                (9) Configuration
          (5) Create file from RAM              (0) User functions

->      Press key in () for desired function: 4

Title: User name and Job number                 Device type: AMD 2732
I/O Format: Binary                              Data File: *
Directory c:\FORTH

                  (1) Enter file name to be loaded
                  (2) Set operation boundaries
                  (3) Select I/O Format
                  (4) Load RAM from current data file
                  (5) Select device type

->      Press key in () for desired function: 1



Current data file: *

Directory of c:\FORTH
 .              ..              NBSBASE.SCR


->      Enter data file and RETURN: lst.scr

Press SPACE to view remaining directory
```

```
Loading RAM from file: 1st.scr

Press CTRL-Z to abort


Operation successfully completed.
          Current sumcheck: 58CD
->      <F1>


Title: User name and Job number
Programmer: Model 29 with UniPak                Programmer size: 16K
Device type: AMD 2732                           I/O Format: Binary
Port: COM1/9600 baud                            User Function File: *
Data File: *                                    Directory: c:\FORTH

           (1) Program device                   (6) Error logging
           (2) Verify device                    (7) File functions
           (3) Load RAM from device             (8) Edit functions
           (4) Load RAM from file               (9) Configuration
           (5) Create file from RAM             (0) User functions


->      Press key in () for desired function: 1



Title: User name and Job number
Programmer: Model 29 with UniPak                Programmer size: 16K
Device type: AMD 2732                           Device Test: Illegal
Parts correctly programmed: 0                   Data File: 1st.scr

           (1) Begin programming                (4) Select blank/
           (2) Select device type               (5) Set up handler
           (3) Set operation boundaries         (6) Begin handler


->      Press key in () for desired function: 1
```

```
Title: User name and Job number
Programmer: Model 29 with UniPak                Programmer size: 16K
Device type: AMD 2732                           Device Test: Illegal
Data File: 1st.scr

Programming device, please wait ...

Press CTRL-Z to abort


->      Press SPACE to program device again  <F1>

Repeat process for:

2nd.scr
3rd.scr
4th.scr
5th.scr
6th.scr
7th.scr
8th.scr
```

Appendix D. PC Compatible AGV Control Software

1. COMPILING AND LINKING

To compile and link the system the following .C files are needed: COMM.C, TEST.C, and GFXPAK.C. They should be compiled and linked using TurboC, from Borland. The .H header files needed are as follows: AGVCOM1.H, AGVCOM2.H, GFXCOM.H, STDIO.H, DOS.H, IBMKEYS.H, ASPORTS.H, and TIMEDATE.H. The first three header files should be in the directory with the .C files, the next two should be in \src\turboc and the last three in \src\ccc. For compiling the files TCCONFIG.TC and TEST.PRJ should be with the *.C files. To do the actual compile and link, enter TurboC from DOS by typing TC <return>. Go to the Compile window by typing Alt-C, and select Build All by typing B. If an error occurs type Alt-P, and make sure the project name is TEST.PRJ. If no project is shown, or it isn't TEST.PRJ, type P then type TEST <return>, after which, type Alt-C and B again. Alt-X will exit you from the TurboC environment.

2. RUN TIME FILES

To run the system the following files should be in the current directory: TEST.EXE, AGVMODE.DAT, MSGSET.DAT, INPUT.DEF, and MSGSET.DEF. The files do the following:

TEST.EXE - The main program.

AGVMODE.DAT - Sets values at run-time for I/O mode, lights enable, keyboard emulation, test speed, agv id, and sonar enable.

MSGSET.DAT - Contains message sets that are read in if the mode is set to file msg I/O.

INPUT.DEF - Holds information for the screen setup.

MSGSET.DEF - Contains the internal message sets that can be executed by certain commands.

MSGSET.DAT and MSGSET.DEF are both generated as described in 3. AGVMODE.DAT can be changed by any editor, and is described in detail in 4. INPUT.DEF can also be changed by any editor and is described with the screen I/O in section 11.

3. MHWSINT.BAT, MHS.BAT, MHS.EXE, AND CREATING MESSAGE SETS

The two batch files, MHS.BAT and MHSINT.BAT, can be used to call MHS.EXE and cause it to create new message set files. Both batch files call MHS.EXE and then return to \agvctl, copying the new message set into the proper file. To create a new internal set file (MSGSET.DEF) type MHSINT <return> from DOS. To create a file I/O message set (MSGSET.DAT) type MHS <return> from DOS. Either will put you into MHS.EXE. This executable file holds a group of message sets. It is a version of the MHS controller modified to send message sets to a file instead of a port. Message sets are numbered from zero to 103, with from 1 to 12 messages in each set. The PGUP and PGDN keys move you from set to set, while the up and down arrows move from message to message. Once in a message, the left and right arrow keys move you from parameter to parameter. The parameters can be changed while using the right and left keys, with a return storing the values. To send a message set to the file the message set name should be located using the up/down arrows and a CNTRL-return entered. As many sets as desired can be stored in this way. When all the sets needed are stored an esc leaves the executable. Re-running the mhs.exe will reinitiallize the file. In MSGSET.DEF (the internal sets) the first message (#0) is ignored, therefore when creating this file hit CNTRL-return twice for the first message set entered. If mhs.exe is run from the batch file, the proper file will get the message sets and you will end in the directory \agvctl.

4. THE RUN TIME MODES

The AGV OS can be set to several different modes during runtime, these modes being set from AGVMODE.DAT. The file looks as follows:

mode	keybd_enable	sonar_enable	lights_enable	test_speed	agv_id
3	1	0	0	1	0

Mode controls the I/O to both the serial port (modem) and the other ports (AGV I/O). The Message I/O can be conducted either over the serial port or from the file MSGSET.DAT. In File mode the messages are started by the keyboard numbers (0 - 9) and read from the file, with no status being sent. In serial mode the message sets and status requests are recieved by the port, with the status being returned through the same port. The input from the AGV can either be read from the AD board and the parallel port, or can be generated internally by the program. The internally generated AGV input allows the keyboard to control On/Off wire and Optics (To have the keyboard inputs faked by the

program KEYBOARD_ENABLE should be set to false (0). See below). Output to the AGV can be disabled if the cart isn't hooked up. This keeps the printer port from being disrupted.

The value for mode for each combination of I/O is listed in this table.

mode#	msg I/O	AGV input	AGV output
0	File	Faked/Keybd	None
1	File	Faked/Keybd	Send
2	File	Real	None
3	File	Real	Send
4	Serial Port	Faked/Keybd	None
5	Serial Port	Faked/Keybd	Send
6	Serial Port	Real	None
7	Serial Port	Real	Send

Keyboard Enable controls the method used to fake the AGV inputs. A 1 allows the keyboard to control Optics and On/Off wire, while a 0 indicates the optics and wire are to be faked internally.

The Sonar Enable and Lights Enable values indicate whether the sonar range should checked or the lights should be flashed, with a 1 being the on and 0 being the off value.

Test Speed controls the speed of command execution in fake execution mode. A 1 is normal speed and a 0 is fast.

Agv Id is the ID number used by the commands to indicate which cart gets which message set. If it doesn't match the command AGV ID numbers the message will be ignored.

5. FILES GENERATED BY THE SYSTEM

The files generated by the system include: DBUG.DAT, LOG.DAT, PORT.DAT, and SCRDMP.DAT. These files are reinitiallized everytime the program is started. To print a value to a file the format is:

fprintf (?????_fp, "YOUR COMMENT",ARG's), with ???? being the name of the file, and the rest of the line working exactly like a printf.

DBUG.DAT - A general use file for tracking and variable value checks. No restrictions on use.

LOG.DAT - Lists the commands as they are put into the message set

and as they are executed. Internal sets are listed as they are used and executed. This file should be used only on on a short term for debugging.

PORT.DAT - All input and output bytes are listed as they are recieved or sent. Short term debugging only. Note: a 6 on input is the Acknowledge signal.

SCRDMP.DAT - Holds all the screens dumped during execution. Shouldn't be used for debugging.

6. AN EXAMPLE OF CONTROL FLOW

An example of how a message set is recieved and executed is provided by the following: The message set being sent to the agv is message set #78 in the MHS.

```
Msg set  0:  TWS to MHS_CA
EE 00 55 09 00 B0 0E
EE 01 15 11 80 B0 57
EE 02 24 04 A3 B0 7D
EE 03 16 08 D3 04 F8
EE 04 26 04 53 07 88
EE 05 3A 67 21 04 CB
```

Each message is recieved byte by byte, with the program checking the checksum before each 7 byte message is stored. An acknowledge is sent after each message. Messages are read in until a message with the last message bit set to On is sent. At that point the current message number is set to zero. Start new command is called. This routine decodes the message according to it's type and sets up the initial values and statuses needed for execution. Start new command also sets the current command status to zero, allowing the current command to be executed. At this point control will pass from get agv status to process current command every cycle until the message is completed. A special case for execution is message type 5. This message type calls an internal set. When it is detected in Start new command the needed internal set is loaded and executed, when it is done control is passed back to the original message set.

7. STATUS VARIABLES

The status variables indicate whether a subroutine, or set of subroutines is being, or should be called. A 0 indicates executing and a 1 indicates idle.

The 0 for a variable indicates the following:

auto_mode_status - Executing a command message that involves moving the cart, either virtually or wire guided.

manual_mode_status -

charge_status - In the process of extending or retracting the charge probe.

extend_status - Extending the probe.

retract_status - Retracting the probe.

roller_status - Activating the roller deck, either load or unload.

first_dist_status - Moving the cart under automatic control the distance set from the command message. Guidance can be either virtual or wire. In the case of virtual the wheel angle is straight.

dist_2_status - Same as first_dist_status, except the angle for virtual movement is set from the message.

last_dist_status - Same as first_dist_status.

optic_count_status - Optics are counted until the number indicated in the message are spotted.

seek_wire - Indicates that at a certain point seek wire status should be turned on.

seek_wire_status - Looking for the wire for three feet. If the wire isn't found stop the message set. If the wire is found change to wire guidance.

stop_cart_status - Indicates that a slow stop is in progress.

e_stop_status - The sonars are being checked.

stop_w_delay_status - The cart is stopped and will wait for two seconds.

The three distance statuses and the optic status are executed in order: first_dist_status, dist_2_status, last_dist_status, then optic_count_status. Optic calls stop cart if the stop_go bit is set to stop, otherwise it ends the command.

8. NON-STATUS VARIABLES

This is a description of many of the global variables.

????_count and ????_cnt - Count variables, so some subroutines are only executed every few cycles.

TACH VARIABLES -

tach_reading - The value read from the AD board from the speed tach.

current_dist - The distance traveled since the last reset (floating).

int_dist - The integer value of current distance.

SCREEN AND SYSTEM VARIABLES -

help_num - The current help line to be displayed.

help_toggle - This indicates that the help_num has been changed.

help_blank - If this is true the help area is left blank.

sonar_enable - A zero here disables the sonar.

lights_enable - Enables the lights to flash (otherwise they stay off).

agv_test_speed - Zero is fast test speed, 1 is normal.

keybd_enable - A one allows keyboard inputs for test mode (Optics and On/Off wire) to be generated by the program (i.e. no operator needed).

key_optic_on - The status of the keyboard generated optic (Broken/Unbroken).

key_e_stop - The status of the keyboard generated sonar stop.

key_wire - Indicates the status of the keyboard generated On/Off wire.

key_back_optic - Indicates that the optic on the wrong side for a tray has been broken by the keyboard(test mode only).

error_num - The current error value. A list is in MSGFMT.IFT type 15.

agv_id - The id number the cart uses to identify messages intended for it.

last_msg_num - The highest message number in the set.

mode - The I/O mode (See #4).

curr_msg_num - The message being executed.

curr_cmd_status - A zero indicates that a command is in progress.

cmd_start_time - The start time of the last command.

message_set [][] - The currently executing message set.

command_msg [] - The message being read in.

status_msg [] - Status message last time it was requested.

msg_set_hold [][] - The external set being held while an internal set is executing.

message_set_buff [][][] - The internal message set buffer.

I/O VARIABLES -

in_out[31] - The array that holds most of the values inputed and outputed by the AD and DA boards.

ad_port - Most of the major bit values to be sent out (Not roller or charge).

parallel_byte - A byte to go out over the parallel port (bit values).

para_port - One of the values to be sent out the parallel port (bit values).

stat - The latest status byte from the parallel port.

ROLLER AND CHARGE VARIABLES -

roller_choice - The roller to be used (Front/Back).

roller_direction - The left_right direction for the roller to go.

roller - Roller deck Enable/Disable.

roller_error_status - The value that indicates which roller execution case statement the deck is currently in.

tray_error - The current error value of the tray (RECOVERABLE/UNRECOVERABLE).

extended, retracted, charge, arm_connected - Charge probe status.

PROGRAM CALCULATED VARIABLES -

left_total, right_total - Filtered values from the wire sensor.

sonar_stopped - Indicates a sonar stop if True.

virtual_set_speed - The speed at which virtual movement is to be executed.

halt - A true indicates that the cart isn't moving. (THIS IS IMPORTANT).

optics_spotted - The number of optics that have completed the beam since the last reset.

INTERNAL MESSAGE SET VARIABLES -

internal_msg_set - A flag meaning the set being executed is internal.

restore_msg_set_flag - The break out flag to get out of start_new_cmmd and start execution of an internal set.

internal_set_last - The number of the last set of a type 5 msg.

internal_set_two - The set number for the second set to be executed (0 is none).

old_msg_num - The saved current external message set.

old_last_msg - The saved value of the last message of the external set.

guidance_type - Wire or Virtual.

set_speed - The speed value decoded from the message.

optic_speed - The speed to go when looking for optics.

optic_count - The number of optics to look for.

stop_go - This indicates whether the cart should stop after the last message in the set, or continue.

fwd_rev - The direction.

asrs_direction - Towards or away from the asrs.

turn_direction - Left or Right. (facing forward from the cart)

load_unload - Which roller operation.

first_dist - The distance to travel in a straight line.

dist_2 - The distance to travel at an angle 'angle'.

last_dist - The other distance to travel in a straight line.

angle - The angle to set the wheel to during distance 2.

9. MODIFYING THE SCREEN

The screen layout is defined by the values in INPUT.DEF, and the subroutine UPDATE_AGV_SCREEN. INPUT.DEF looks as follows: Note: The text cannot contain embedded spaces.

GRP	ROW	COLUMN	LENGTH	INT_VALUE	TEXT_WIDTH	TEXT
0	4	16	1	0	15	Dead_Micro
0	5	16	1	0	15	Sonar_Stopped
0	6	15	2	0	14	Error_Number
1	8	16	5	0	15	Rev_Relay
1	9	16	1	0	15	Fwd_Relay
1	10	16	1	0	15	Brake_On/Off

2 1 15 0 0 0 NBS_AGV_Control_by_Interface_Technology_Inc.

The data from this file is read in at the beginning of the program and used to update the screen. The data has 6 integer and one string field. The first field is the group number. The values to be put on the screen are divided into groups, with a maximum of MAX_GROUPS groups, and MAX_PARMS values per group. The groups should be in order, starting with group 0. The next two fields are the row and column at which the leftmost digit will be located. Length is the size of the field allocated for the value. Init val is the initial value put in the screen location. Text width is the number of spaces left of the row-column value the text is started. The text string is put on the screen at the location figured by row-column and width. The string can have no embedded spaces, so underscores _ in the string are replaced with spaces on the screen.

The row, column, length, group, and value of each location is stored in an array. UPDATE_AGV_SCREEN calls UPDATE_SCR_FIELD for each variable to be updated on the screen. Each variable to be checked is compared to the value indicated on the screen, if they don't match the screen is updated. To put an integer variable value on the screen the variable needs to be passed to UPDATE_SCR_FIELD using the following format:

update_scr_field (&??????, GROUP NUM, NUMBER INSIDE GROUP);
????? is the variable name, and the group number and number inside the group are the array number from INPUT.DEF. For example, to put the value of a variable named error_num next to the words "Error Number", with the words starting at location (6, 1) and a field width of two for the value; using the INPUT.DEF above, the call would look like this:

```
update_scr_field (&error_num, 0, 2);
```

UPDATE_AGV_SCREEN also updates the help lines if needed. The help lines are listed in a case statement, to add a help line just add a new case, and set the help_num to the new value whenever the line needs to be up and to zero when it is done. Help_toggle should also be set to TRUE whenever the help_num is changed.

Appendix E. ASRS Command and Status Message Formats

The general command message format is,

<Header><Unit><Command><Argument><Terminator>

The header field is one byte with a value of 1. The unit (or module) field is one byte which has a value of 0 to 255. The value for each module is shown in the following table.

Module No.	Unit Value
1	0
2	4
3	8
4	16
5	32
6	36

The command field is one byte representing a command function to be performed. Numerous commands are available to check out and adjust the system in addition to the tray transfer comands. The commands used by the MHS control system are detailed in the following table. The command column shows the value of the comand byte in hex.

Command	Function
43	Transfer tray from specified shelf to conveyor.
44	Return tray to specified shelf from conveyor.
4B	Transfer tray from ASRS to AGV.
4C	Transfer tray from AGV to ASRS.
4E	Update system status.
1B	Send status to MHS controller.

The argument field is not required for all commands, but it is required for the transfer tray commands shown in the above table. This field usually contains one or more numeric ASCII characters. For example, for the transfer tray commands, if the shelf number is 13, the field would contain two bytes; 31 and 33 hex. The terminator field is 1 byte which usually has a value of 0A hex.

The system status is a one byte message described as follows:

```
Status bit 0 = S1      - Error code. See below.
       bit 1 = S2      - Error code. See below.
       bit 2 = S3      - Error code. See below.
       bit 3 = Unknown - Tray storage location unknown.
       bit 4 = Error   - Failed to complete command function.
       bit 5 = Busy    - System is busy performing a function.
       bit 6 = Host    - The host buffer contains data.
       bit 7 = S/flag  - Marks this character as status.  Always
                         set.
```

If the Error bit 4 is set S1, S2, and S3 are interpeted as follows:

S1	S2	S3	
0	0	1	- Loc Full, a shelf location was found to be full.
0	1	0	- Loc Empty, a shelf location was found to be empty.
0	1	1	- In Use, requested tray is in use at another delivery location.
1	0	0	- Height, tray contents to high for storage location.
1	1	0	- Illegal shelf, shelf position not valid for unit or has been canceled.
1	1	1	- Conveyor, error related to conveyor.

Appendix F. Roller Table Command and Status Formats

The roller table controllers accept a 1 byte alphabetic character as shown in the following table.

Command	Function
j	Give up control of table 1.
k	Take control of table 1.
l	Give up control of table 2.
m	Take control of table 2.
o	Transfer table 1 tray from table to AGV
p	Transfer table 2 tray from table to AGV
q	Transfer table 1 tray from AGV to table
r	Transfer table 2 tray from AGV to table
s	Send latest status
t	Send tray position status

The status message consists of 8 bytes, 1 B C 2 E F CR LF. Bytes B and E designate whether the workstation has taken control of trays 1 and 2 respectively. '0' indicates it has not, and '1' indicates that it has.

C and F are the status of trays 1 and 2 respectively as follows:

C & F	Status
0	No tray on table
1	Tray is currently on table
2	Tray is between AGV and stop point
3	Error, tray handler is in local mode
4	Error, no tray motion
5	Loading in progress
6	Unloading in progress
7	Error, improper initial position
8	Error, tray did not transfer to AGV
9	Error, tray did not reach final position
A	Error, illegal switch readings

Appendix G - Material Handling System Work Elements

1. Workstation Level

DELIVER_TRAY

```
WS_ID = <MHS>
PLAN_ID = <process plan operation sheet identifier>
PLAN_VERSION = <version number of process plan>
TRAY_ID = <logical tray identifier>
TRAY_TYPE = <tray configuration type>
ITEM_SER_NR = <tray serial number>
LOT_ID = <current lot associated with tray>
FROM = <pickup cart stop>
TO = <dropoff cart stop>
```

Note: This is a complex work element that is decomposed into tasks to be executed by the MHS equipment level controllers. Work elements appearing in the reference PLAN_ID, which may be used to decompose a DELIVER_TRAY, include:

GOTO, TRANSFER_TRAY, GOTHRU, GET_TRAY, PUT_TRAY

2. Equipment Level

GOTO

```
      AGV_ID = <AGV_1 or AGV_2>
 ROLLER_BED = <FRONT or BACK>
 FROM_POINT = <a point immediately ahead of TO_POINT>
   TO_POINT = <a point immediately after FROM_POINT>
```

TRANSFER_TRAY

```
      AGV_ID = <AGV_1 or AGV_2>
FROM_ROLLER = <FRONT, BACK or a transfer point>
  TO_ROLLER = <FRONT, BACK or a transfer point>
```

GOTHRU

```
      AGV_ID = <AGV_1 or AGV_2>
 FROM_POINT = <cart stop point or through points>
 THRU_POINT = <cart through points>
```

```
GET_TRAY

       UNIT_ID = <ASRS_1 - ASRS_6 or MBD_1 - MBD_2>
       TRAY_ID = <serial number of tray or NA for ASRS>
      SHELF_ID = <shelf number of tray or NA for MBD>

PUT_TRAY

       UNIT_ID = <ASRS_1 - ASRS_6 or MBD_1 - MBD_2>
       TRAY_ID = <serial number of tray or NA for ASRS>
      SHELF_ID = <shelf number of tray or NA for MBD>

CHARGE_BATTERY

        AGV_ID = <AGV_1 or AGV_2>
      CHG_TIME = <charge time in minutes>

RETRACT_PROBE

        AGV_ID = <AGV_1 or AGV_2>


Cart through points:

     CWS_STA
     TWS_STA
     VWS_STA
     HWS_STA
     IWS_STA
     MHS_ASRS
     MHS_HB

Cart stop points:

     MHS_CA - Cart charging area
     MHS_IP - Tray inspection point
     MHS_TPn - ASRS transfer point where n = UNIT_ID
                  <1, 2, 3, 4, 5, or 6>
     CWS_TPn  (n = 1 or 2)
     IWS_TPn  (n = 1 or 2)
     TWS_TPn  (n = 1 or 2)
     VWS_TPn  (n = 1 or 2)
     HWS_TPn  (n = 1 or 2)

Shelves:

     SHELF_ID = SHELF_n where n = 1 to maximum number of shelves
```

READER COMMENT FORM

Document Title Material Handling Workstation Implementation

This document is one in a series of publications which document research done at the National Bureau of Standards' Automated Manufacturing Research Facility from 1981 through March, 1987.

You may use this form to comment on the technical content or organization of this document or to contribute suggested editorial changes.

Comments: _____

_____

_____

_____

_____

_____

_____

If you wish a reply, give your name, company, and complete mailing address: _____

_____

_____

_____

What is your occupation? _____

NOTE: This form may not be used to order additional copies of this document or other documents in the series. Copies of AMRF documents are available from NTIS.

Please mail your comments to: AMRF Program Manager
National Bureau of Standards
Building 220, Room B111
Gaithersburg, MD 20899

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 88-3784	2. Performing Organ. Report No.	3. Publication Date MAY 1988

4. TITLE AND SUBTITLE

Material Handling Workstation Implementation

5. AUTHOR(S)

Carl E. Wenger

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

The purpose of this document is to provide a general description of design and implementation of the AMRF Material Handling Workstation (MHWS). The MHWS equipment includes two Automatic Guided Vehicles (AGVs), an Automatic Storage and Retrieval System (ASRS), and roller tables at other workstations. The document should provide the reader with an understanding of concepts used to implement the MHWS.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

AMRF; implementation; manufacturing, Material Handling Workstation

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

54

15. Price

$13.95